Übungsblatt 3

OPR2UE, Objektorientierte Programmierung, UE

Erreichbare Punkte	36 Punkte
Letzter Abgabetermin	Montag, 8. Mai 2017, 23:55 Uhr
Abgabedokumente	Quelldateien inkl. Tests
Abgabeort	Moodle (https://hagenberg.elearning.fh-ooe.at)

Aufgabe 1 - Punkte und Dreiecke 16 Punkte

a) Punkte 4 Punkte

Entwerft eine Klasse, welche einen Punkt in der kartesischen Koordinatenebene repräsentiert. In der Klasse sollen die folgenden Methoden realisiert werden:

- `Point(double x, double y)` - Konstruktor
- `getX()` – liefert die horizontale Koordinate des Punkts
- `getY()` – liefert die vertikale Koordinate des Punkts
- `distance(Point p)` – liefert den Abstand zwischen p und der jeweiligen Instanz[1]
- `isSame(Point p, double within)` – stellt fest, ob die Instanz mit dem gegebenen Punkt übereinstimmt (die Abweichung darf maximal den Wert von `within` betragen)
- `moved(double x, double y)` – liefert einen neuen Punkt, welcher horizontal und vertikal verschoben ist.

b) Dreiecke 8 Punkte

Entwickelt eine Klasse, welche Dreiecke in der kartesischen Koordinatenebene abbildet. Für die Umsetzung soll die in Aufgabe 1 erstellte Klasse `Point` weiterverwendet werden. Folgende Methoden sollen von der Klasse implementiert werden:

- `Triangle(Point a, Point b, Point c)` - Konstruktor
- `perimeter()` – liefert den Umfang des Dreiecks
- `area()` – liefert die Fläche des Dreiecks[2]
- `isSame(Triangle t, double within)` – stellt fest, ob die Instanz mit dem gegebenen Dreieck übereinstimmt (unter Berücksichtigung der gegebenen Toleranz `within`)
- `moved(double dx, double dy)` – erzeugt ein vertikal und horizontal verschobenes Dreieck
- `zoomed(double f)` – erzeugt ein um den Faktor f gestrecktes Dreieck; Zentrum der Streckung ist der Koordinatenursprung

c) Tests 4 Punkte

Testet die in Aufgabe 1 und 2 entwickelten Klassen mit Hilfe von JUnit-Tests.

---

[1] Zur Berechnung des Abstands kann die Methode **`Math.hypot()`** verwendet werden.
[2] Zur Berechnung der Fläche kann die Methode **`Math.sqrt()`** verwendet werden.

Aufgabe 2 - ArrayList 20 Punkte

a) Implementierung 10 Punkte

Eure Aufgabe ist es, eine Klasse `ArrayList`, welche das auf Moodle zur Verfügung gestellte `List`-Interface implementiert, zu erstellen.

Die Klasse `ArrayList` soll über folgende Funktionalitäten verfügen:

- Rückgabe der Referenz auf das jeweils erste bzw. letzte Element der Liste
- Rückgabe der Anzahl der Elemente in der Liste
- Hinzufügen eines Elements am Ende der Liste
- Rückgabe eines einzelnen Elements der Liste, welches über den Index des Elements angegeben wird; wird ein falscher Index angegeben, so wird `null` zurückgegeben
- Löschen von Elementen in der Liste, welche den gleichen Wert haben (durch das Löschen dürfen keine Lücken in der Liste entstehen)
- Leeren der gesamten Liste
- Rückgabe einer String-Repräsentation der Inhalte der Liste

Die Implementierung soll dabei folgende Eigenschaften aufweisen:

- Die Elemente werden in einem Array verwaltet. Die initiale Größe dieses Feldes ist 4.
- Wird die Kapazität des Arrays zu klein, so soll diese automatisch verdoppelt werden. Ein Verkleinern des Feldes, wenn wieder Elemente entnommen werden, ist nicht zu implementieren.

Hilfsmethoden zur Umsetzung der Anforderungen können selbstverständlich eingesetzt werden. Die ursprünglichen Interfaces dürfen jedoch nicht verändert werden.

b) Deque 6 Punkte

Zusätzlich zu den zuvor beschriebenen Funktionalitäten soll die `ArrayList` auch als *Deque* (double ended queue) verwendet werden.

Entwickle auch hier das geeignete Interface `Deque` und implementiere dieses wiederum in `ArrayList`. `Deque` muss folgende Aktionen unterstützen:

- Hinzufügen von Elementen sowohl am Beginn als auch am Ende der Liste
- Entfernen von Elementen sowohl am Beginn als auch am Ende der Liste
- Zurückgeben des ersten bzw. des letzten Elements in der Liste

Achte bei der Implementierung darauf, Codeverdoppelungen zu vermeiden. Greife auf bereits bestehende Funktionalitäten zurück.

c) Tests 4 Punkte

Um sicherzugehen, dass die implementierte `ArrayList` auch wirklich das gewünschte Verhalten realisiert, muss sie eingehend getestet werden.